2. Jelaskan definisi Formal dari terminology berikut : **(Point 20)**
   a. Graph
   b. Tree
   c. Binary Tree
   d. Internal Node
   e. External Node
3. Buatlah program dalam bahasa lisp secara recursive, untuk membalik suatu list linear, **TANPA** menggunakan fungsi **REVERSE atau INVERSE** yang sudah terdefinisi. Definisi dan Spesifikasi fungsi sebagai berikut : **(Point 30)**

```
Function Balik : List1, List2 → L1 yang terbalik
/*L1 adalah list linear boleh kosong
  L2 selalu bernilai NIL
*/
Aplikasi :
(revreccur '(1 2 3) nil)
→(3 2 1)
(revreccur '(1 (2) 3) nil)
→(3 (2) 1)
(revreccur '(1 (2) (3 4)) nil)
→((3 4) (2) 1)
```

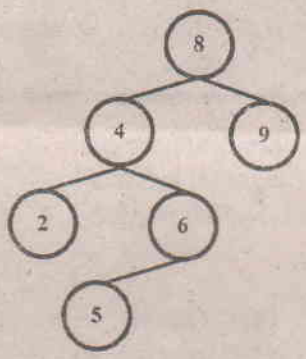
Soal yang berhubungan dengan pohon biner mengacu pada kamus d bawah ini

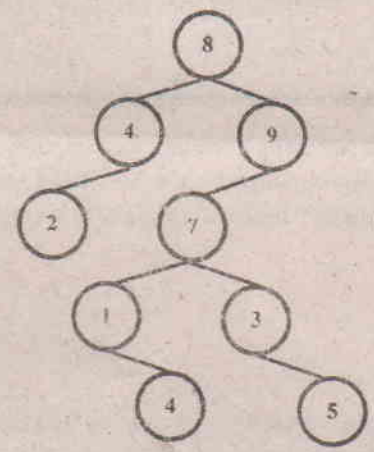
type Elemen : integer
type PohonBiner : < A : Elemen, L : PohonBiner, R : PohonBiner > {notasi prefix }
DEFINISI DAN SPESIFIKASI SELEKTOR
Akar : PohonBiner tidak kosong → Elemen
{ Akar(P) adalah Akar dari P. Jika P adalah //L A R\\ = Akar(P) adalah A }
Left : PohonBiner tidak kosong → PohonBiner
{ Left(P) adalah sub pohon kiri dari P. Jika P adalah //L A R\\, Left (P) adalah L }
Right : PohonBiner tidak kosong → PohonBiner
{ Right(P) adalah sub pohon kanan dari P. Jika P adalah //L A R\\,Right (P) adalah R}

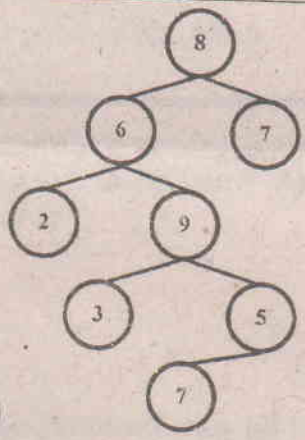
DEFINISI DAN SPESIFIKASI PREDIKAT
IsTreeEmpty : PohonBiner → boolean
{ IsTreeEmpty(P) true jika P kosong : (// \\) }
IsOneElmt : PohonBiner → boolean
{ IsOneElement(P) true jika P hanya mempunyai satu elemen, yaitu akar (// A \\) }
IsUnerLeft : PohonBiner → boolean
{ IsUnerLeft(P) true jika P hanya mengandung sub pohon kiri tidak kosong: (//L A \\) }
IsUnerRight : PohonBiner → boolean
{ IsUnerRight (P) true jika P hanya mengandung sub pohon kanan tidak kosong: (//A R\\) }
IsBiner : PohonBiner tidak kosong → boolean
{ IsBiner(P) true jika P mengandung sub pohon kiri dan sub pohon kanan : (//L A R\\) }
IsExistLeft : PohonBiner tidak kosong → boolean
{ IsExistLeft (P) true jika P mengandung sub pohon kiri }
IsExistRight : PohonBiner tidak kosong → boolean
{ IsExistRight(P) true jika P mengandung sub pohon kanan }
Max S:deret bilangan tidak kosong → integer
{ Max (S) menghasilkan nilai elemen (atom) yang maksimum dari S }

4. Buatlah suatu predikat secara rekursif untuk mengetahui apakah terdapat *path* dalam *S* yang berjumlah *n* dimana *S* adalah ***list of list integer*** dan *n* adalah *integer* >= *0*. Berikut definisi dan spesifikasi predikatnya : (**Point 30**)
Function IsN-There : n, S→Boolean
{ **IsN-There** (n, S) mengembalikan **True** jika terdapat **path** dalam S yang berjumlah **n**, selain itu **Nil**, S kosong mengembalikan **Nil** }
Contoh Aplikasi:

(8 (4 (2) (6 (5))) (9))	(8 (4 (2)) (9 (7 (1 (4))(3 (5)))))	(8 (6 (2) (9 (3) (5 (7))))







n=8+4+2=14
n=8+4+6+5=23
n=8+9=17

n=8+4+2=14
n=8+9+7+1+4=29
n=8+9+7+3+5=32

n=8+6+2=16
n=8+6+9+3=26
n=8+6+9+5+7=35
n=8+7=15

```
(IsN-There 14 '(8 (4 (2) (6 (5))) (9)))
→T
(IsN-There 23 '(8 (4 (2) (6 (5))) (9)))
→T
(IsN-There 17 '(8 (4 (2) (6 (5))) (9)))
→T
(IsN-There 14 '(8 (4 (2)) (9 (7 (1 (4)) (3 (5))))))
→T
(IsN-There 29 '(8 (4 (2)) (9 (7 (1 (4)) (3 (5))))))
→T
(IsN-There 32 '(8 (4 (2)) (9 (7 (1 (4)) (3 (5))))))
→T
(IsN-There 16 '(8 (6 (2) (9 (3) (5 (7)))) (7)))
→T
(IsN-There 26 '(8 (6 (2) (9 (3) (5 (7)))) (7)))
→T
(IsN-There 35 '(8 (6 (2) (9 (3) (5 (7)))) (7)))
→T
(IsN-There 15 '(8 (6 (2) (9 (3) (5 (7)))) (7)))
→T
```